## Amendments to the Specification:

Please replace paragraph 6 at page 5, lines 26 through page 6, line 4 with the following amended paragraph:

Referring to Fig. 1, an overview of a multi-processing data processing system ~~150~~ is depicted. For clarity and ease of presentation, an eight-processor system has been depicted. As will be readily appreciated by those of ordinary skill in the art, the invention is applicable to multi-processor systems having other numbers of processors. It is also not necessary that each processor group have four members, or that all processor groups have the same number of processors.

Please replace paragraph 2 at page 7, lines 5 - 17 with the following amended paragraph:

Looking again at Fig. 1, the various memory components within system ~~150~~ can be conceptualized as comprising three processing levels. Each level can be considered to contain one or more "instances" or nodes in the CPU/cache/shared memory hierarchy. Level 0 contains eight level 0 instances, or in the case of four locales or blocks, sixteen level 0 instances, each instance being one of the CPUs 100-107 and its associated caches 108. Level 1 contains two level 1 instances, each comprising four level 0 instances, and a secondary cache. While eight processors are shown, the system can be arranged with more processors, for example, as a sixteen processor system. In the case of a sixteen processor system, level 1 contains four level 1 instances, each confirming four level 0 instances and a secondary cache. Finally, there is a single level 2 instance containing the two level 1, or as applicable, the four level 1, instances and the shared system memory.

Application No. 09/677,120
Amendment dated June 23, 2004
Reply to Office Action dated March 25, 2004
Express Mail EV406652085US

Please replace paragraph 4 at page 7, lines 27 through page 8, line 14 with the following amended paragraph:

More specifically, a job will take the least amount of time if it is always run on the same processor. This is because of the multiple levels of memory caches used to reduce memory latency. In general, the lower the cache level, the less time it takes to access the memory. When a job is run on a processor, its working set of memory is loaded into the applicable cache 108 of one of processors 100-107. On a NUMA system, there is also another level of cache 110 or 111 that is shared between the four processors 100-103 or 104-107 of a functional block. In the best case scenario, the job is run on the processor which has the required code and data in its cache. The next best case scenario is to run the job on another processor on the same NUMA block. The tradeoff in this circumstance is between waiting for the processor on which the job last ran becoming free, or running the job on another idle processor. If time is allowed for the last processor, the idle processor's cycles will be wasted. If the job is run on a different processor, time and bus bandwidth may be wasted filling the cache of the idle processor and pulling all of the modified memory from the cache of the original processor. As noted previously, although the system and method are being described with reference to two blocks or locales 0 and 1, it will be appreciated that it can be implemented with, for example, sixteen CPUs, namely CPUs 0-3 in a locale 0, CPUs 4-7 in a locale 1, CPUs 8-11 in a locale 2, and CPUs 12-15 in a locale 3.

Please replace paragraph 2 at page 8, lines 15 - 21 with the following amended paragraph:

Turning now to the specific disclosure in the drawing, each processor 100-107 keeps track of the time from when it first went busy. It is intended that a process is poached when the processor on which it is scheduled is too busy to run the process. Accordingly, once a predetermined amount of time has elapsed during which a processor continues to be busy, other processors who have gone idle are then free to ~~pouch~~ poach the process from the busy processor. Each processor 100-107 has a timer for this purpose.

Page 3 of 13